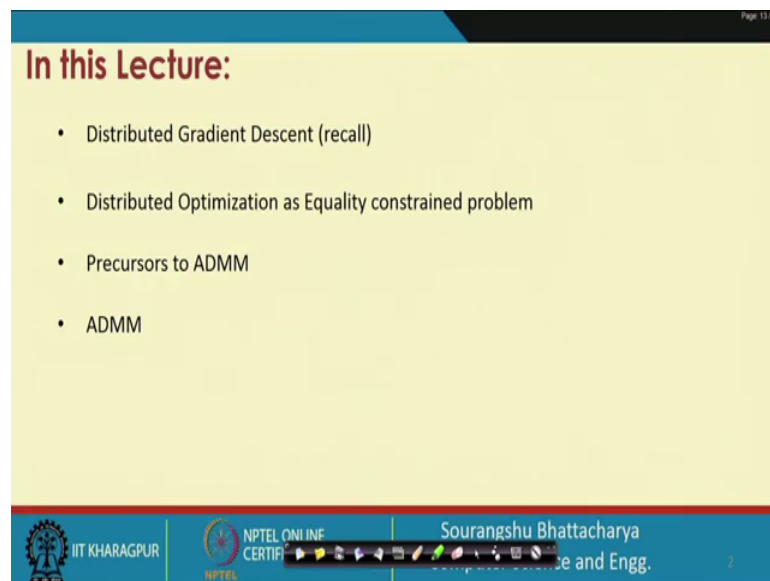


Scalable Data Science
Prof. Sourangshu Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 22 b
Alternating Direction Method of Multipliers (Contd.)

Hello everyone, welcome to the 22nd lecture, second part of 22nd lecture on NPTEL course on scalable data science. I am Prof. Sourangshu Bhattacharya from Computer Science and Engineering at IIT, Kharagpur. And today's lecture is going to be a continuation about the alternating direction method of multipliers or ADMM. So, we in the first part of this lecture, we had looked at the distributed optimization problem and we had looked at how we can pose this as an equality constraint problem or rather equality constrained optimization problem.

(Refer Slide Time: 00:57)



The slide is titled "In this Lecture:" and contains a bulleted list of topics. The background is light yellow with a blue header and footer. The footer includes the IIT Kharagpur logo, NPTEL Online Certification logo, and the name "Sourangshu Bhattacharya" followed by "Department of Computer Science and Engg." and a small number "7".

- Distributed Gradient Descent (recall)
- Distributed Optimization as Equality constrained problem
- Precursors to ADMM
- ADMM

And then we had gone into the precursors of ADMM or the method that is led to ADMM which are basically methods of solving equality constrained optimization problem. So, we have seen the dual ascent method, and we have seen that the dual ascent method has a problem of a problem if the Lagrangian is not strictly convex that is if it does not have a unique optimum then the solution can oscillate.

To solve that problem there is the method of multipliers method which basically uses augmented Lagrangian instead of regular Lagrangian which basically is always strictly

convex. And then finally we have seen that this has a decomposition problem because we use the augmented Lagrangian. So, the method the objective does not directly decomposed which leads to the method of ADMM.

(Refer Slide Time: 02:23)

Alternating direction method of multipliers

- ADMM problem:

$$\min_{x,z} f(x) + g(z)$$
 subject to: $Ax + Bz = c$
- Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$
- ADMM:

$$\begin{cases} x^{k+1} = \operatorname{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} = \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{cases}$$

Handwritten red annotations on the slide: $f_1(x) + f_2(z)$ is written above the objective function. Red arrows point from the x and z terms in the ADMM update equations to the corresponding terms in the Lagrangian. A red bracket underlines the squared term in the Lagrangian.

So, just to recall the ADMM, this is the problem formulation. So, we are using the simplified problem formulation where you have just for simplicity sum of two functions f and g . And these are these are optimized with respect to x and z subject to this kind of an equality constraint.

So, here we use a general equality constraint. We will see in a minute for the specific constraint where x is equal to z which corresponds to the distributed optimization problem ok. And then we have already defined the augmented Lagrangian like this. And we see that because of this term this squared term this actually does not decompose into the form some let us say f_1 of x plus f_2 of z .

So, this is what we would ideally want but it does not happen. But none the less we are we are so the ADMM method can circumvent this by using these three steps. So, in the first steps actually here the problems are actually decomposing. So, in the first step, you only minimize with respect to x keeping the z and the y same for the previous iterate; then you minimize with respect to z , and then finally, you minimize with respect to y ok.

And if you do this, then as you can understand you can do the first one in a different machine, the second one in a different machine and then finally, maybe the third one in a centralized machine, so that is our strategy for distributed optimization. Now, the question is so how did we arrive at this ok.

(Refer Slide Time: 04:52)

Alternating direction method of multipliers

- Problem with applying standard method of multipliers for distributed optimization:
 - 1) there is no problem decomposition even if f is separable.
 - 2) due to square term $\frac{\rho}{2} \|Ax - b\|_2^2$
- The above technique reduces to method of multipliers if we do joint minimization of X and z
- Since we split the joint X, Z minimization step, the problem can be decomposed.

So, basically we see that the; so we have seen that since we split the joint x, z minimization problem, now the solution can be decomposed ok.

(Refer Slide Time: 05:35)

Alternating direction method of multipliers

- ADMM problem:

$$\min_{x,z} f(x) + g(z)$$
 subject to: $Ax + Bz = c$
- Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$
- ADMM:

$$\begin{aligned} x^{k+1} &= \arg\min_x L_\rho(x, z^k, y^k) \\ z^{k+1} &= \arg\min_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

(Handwritten note: $(x^{k+1}, z^{k+1}) = \arg\min_{x,z} L_\rho(x, z, y^k) \rightarrow M015$)

So, earlier if we were not using ADMM, the first step of our solution would be something like we would have to combine these two steps into saying that x^{k+1} comma z^{k+1} is equal to $\arg \max$ over jointly x and z and then we would have to say something like L rho of x comma z comma y^k . So, this would be the method of multipliers solution ok, instead we can now split it in the ADMM step ok.

(Refer Slide Time: 06:20)

ADMM Optimality conditions

- Optimality conditions (differentiable case):
 - Primal feasibility: $Ax + Bz - c = 0$
 - Dual feasibility: $\nabla f(x) + A^T y = 0$ and $\nabla g(z) + B^T y = 0$
- Since z^{k+1} minimizes $L_\rho(x^{k+1}, z, y^k)$:

$$0 = \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c)$$

$$= \nabla g(z^{k+1}) + B^T y^{k+1}$$
- So, the dual variable update satisfies the second dual feasibility constraint.
- Primal feasibility and first dual feasibility are satisfied asymptotically.

Handwritten notes on the slide:

- $\nabla_y d_\rho(n, z, y) = 0$
- $\nabla_n d_\rho(n, z, y) = 0$
- $\nabla_z L_\rho(x^{k+1}, z, y^k) = 0$
- $y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$

So, now why does this basically work? So, so first some of the some of the simple things, so for this problem to have an or rather for the given solution to be optimal that is optimality conditions ok. So, there are three optimality conditions now, because there are three variables.

So, one is the primal feasibility which is basically corresponding to gradient of L rho L rho with respect to so will with respect to the dual variable which is in our case y ok. So, x, z, y this should be equal to 0. So, this is the primal feasibility condition which also boils down to saying that your constant should be satisfied.

Then there are two dual feasibility conditions ok. One when we differentiate with respect to each of the primal variables, so when we differentiate with respect to x L rho of x, z and y is equal to 0 and once when we differentiate with respect to z . So, these are the three optimality conditions.

Now, as we have seen earlier since z^k minimizes this quantity if we set the gradient of so if we set the gradient with respect to z and we put x^{k+1} and y^k equal to 0, we get this condition. So, this is equal to 0. And now if we just substitute back the update for the y if you recall the update for the y^{k+1} is nothing but your y^k and then plus rho times $A x^{k+1} + B z^k - c$.

So, if you recall, this is your update for the y^{k+1} . And now you can see that it automatically satisfies the one of the dual feasibility conditions, which is this one. So, your z^{k+1} automatically satisfies the second dual feasibility condition. This we saw in case of method of multipliers also just by virtue of using rho as the step size.

So, basically we are left with these two unsatisfied dual feasibility conditions or not dual feasibility, but these two unsatisfied conditions. So, this should be 0, and this should be 0. If these two conditions are satisfied, then we can say that we have already reached the optimal.

(Refer Slide Time: 10:06)

ADMM Optimality conditions

- Primal residual: $r^k = \|Ax^k + Bz^k - c\|$
- Since x^{k+1} minimizes $L_\rho(x, z^k, y^k)$:

$$0 = \nabla f(x^{k+1}) + A^T y^k + \rho A^T (Ax^{k+1} + Bz^k - c)$$

$$= \nabla f(x^{k+1}) + A^T (y^k + \rho^{k+1} + \rho B(z^k - z^{k+1}))$$

$$= \nabla f(x^{k+1}) + A^T y^{k+1} + \rho A^T B(z^k - z^{k+1})$$

or,

$$\rho A^T B(z^k - z^{k+1}) = \nabla f(x^{k+1}) + A^T y^{k+1}$$

- Hence, $s^{k+1} = \rho A^T B(z^k - z^{k+1})$ can be thought as dual residual.

Handwritten notes on the slide:

- $A x^k + B z^k - c = 0$
- $\nabla f(x^{k+1}) + A^T y^{k+1} = 0$
- $y^{k+1} = y^k + \rho \delta^{k+1}$
- $(z^k - z^{k+1}) \rightarrow 0$

So, these conditions are satisfied asymptotically. So, what do we mean by that? So basically in order to monitor the satisfaction of these conditions, now, we define the two residuals. The first one is the primal residual, which is basically trying to monitor the convergence of the rather the satisfaction of the primal feasibility condition, which is the derivative with respect to the dual variable should be 0.

And that is simply this value $A x$, we know that at optimality $A x + B z - c$ should be 0. And if we take r_k to be just this quantity, actually we should take it to be just norm of this quantity ok, then it should be 0. So, if a vector is 0, its norm should also be 0 ok. And the second thing is basically the, we have to monitor the other condition, which is this one ok.

So, this is the first dual feasibility condition that we get that is gradient of $f(x_k) + 1$ plus $A^T y_k + 1$ should be equal to 0 ok. And we see that so if we just now impose the condition that $x_k + 1$ minimizes this quantity ok. And if we take the gradient and equate it to 0, because $x_k + 1$ minimizes this. So we get that this should be equal to 0. And just like rearranging previously ok. So, if we do the rearrangement, so we get that gradient of $f(x_k) + 1$ plus $A^T y_k + \rho^T$ this whole quantity is equal to 0.

Now, we put this term we couple this with the second term. So, we couple the two together. And then we apply the same update here and we see that this is basically the $y_k + 1$ update ok. So, so we can see that your $y_k + 1$ update is basically your $y_k + \rho$ into yeah, so this is yeah so this is your primal residual which is $r_k + 1$ ok. If you recall, this is this is the update, and so this becomes $y_k + 1$ ok.

And now this quantity is the quantity that you want to monitor ok. So, this is nothing but just negative of this, so this quantity is nothing but the negative of this quantity. So, basically monitoring this; sorry monitoring this second primal residual boils down to just monitoring, the difference between z and z_k ok. We will see this mode, but essentially what this tells us is that now this quantity ρ into $A^T B$, A and B are all given. And z_k is the z variable value in the k th time point and z_{k+1} is the z variable value in the $k+1$ th time point.

So, this is the new z , then this is the old z . And if you take the difference between the new z and the old z , and then multiply by this constant, this constant is just known ok. So, if this is if this z minus z_k , so if this is 0 ok. If this goes to 0, then this product will also go to 0. So, basically the other residual monitoring the other residual is equivalent to monitoring that the between two successive iterations the z variable, which is the second primal variable converges and does not basically change ok. So, this can be thought of as the dual residual ok.

(Refer Slide Time: 15:34)

Convergence of ADMM

- Assumption 1: Functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ and $g: \mathbb{R}^m \rightarrow \mathbb{R}$ are closed, proper and convex.
 - Same as assuming $\text{epi } f = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} \mid f(x) \leq t\}$ is closed and convex.
- Assumption 2: The unaugmented Lagrangian $L_0(x, y, z)$ has a saddle point (x^*, z^*, y^*) :
$$L_0(x^*, z^*, y) \leq L_0(x^*, z^*, y^*) \leq L_0(x, z^*, y^*)$$

The slide includes a 3D plot of a saddle point surface with handwritten red annotations. The plot shows a surface that is concave in one direction and convex in another. Handwritten labels include (x^*, z^*) and (x, z) with arrows pointing to the corresponding axes. The bottom of the slide features the IIT Kharagpur and NPTEL Online Certificates logos, along with a small video feed of the presenter.

So, now so we will state now the theorem of convergence of ADMM ok. So, we have already shown the iterates. Now, suppose the function f is f and g are two functions ok, and these are closed proper and convex. So, many of the machine learning a loss functions are of this type ok.

If this is the case, and the second condition is that the augmented Lagrangian has a saddle point that is you have the optimal solution x^* , z^* , and y^* . And you have the, you have the dual problem, which is which is lower than which is basically and upper bound. So, basically if this so this is the convex direction, so and this is the concave direction.

So, if there are two directions ok, and in one direction at this point x^* , z^* , so this is the optimal point x^* , z^* , and then y^* ok. If you go in the x , z direction ok, you always you always go higher. So, if your x is, if you move from x^* and z^* that is you move on this axis you always go higher. And if you move in the y axis, you always go lower. So, this is basically in some sense the dual problem.

And this is the primal problem for some given so the primal problem for some given dual variable value, and is the dual problem for some give given primal variable value. If this condition holds and this condition holds for all convex optimization problems; if this is the case, then your algorithm ADMM algorithm is guaranteed to converge that is that is the basically the k ok.

(Refer Slide Time: 18:15)

Convergence of ADMM

- Primal residual: $r = Ax + Bz - c$
- Optimal objective: $p^* = \inf_{x,z} \{f(x) + g(z) \mid Ax + Bz = c\}$
- Convergence results:
 - 1) Primal residual convergence: $r^k \rightarrow 0$ as $k \rightarrow \infty$
 - 2) Dual residual convergence: $s^k \rightarrow 0$ as $k \rightarrow \infty$
 - 3) Objective convergence: $f(x) + g(z) \rightarrow p^*$ as $k \rightarrow \infty$
 - 4) Dual variable convergence: $y^k \rightarrow y^*$ as $k \rightarrow \infty$

Handwritten note: $s^k = \rho A^T B (z^k - z^{k-1})$

Page 23/27

IIT KHARAGPUR NPTEL ONLINE CERTIFICATE

So, now in practice what will you do, so how will you observe ok? So, we have defined the primal residual already, of course this is the objective function value of a given solution ok. So, this is the rather the primal objective at optimality. So, as the algorithm converges. So, the previous result states that if those conditions are satisfied that is if you are functions are closed proper and convex.

And if it if the Lagrangian has an has a saddle point that is with respect to the dual variable the objective function value always decreases, and with respect to the primal variable values, the objective function value always increases. Then you have that there is convergence that is the iterates will converge, but this result says what the convergence should be so it says that as your k tends to infinity.

So, as you iterate more and more the dual the primal residual will go to 0 ok. The dual residual, which is the s^k that we have defined that, will go to 0. So, recall that s^k is just defined as this $\rho A^T B (z^k - z^{k-1})$ ok. The objective function value $f(x) + g(z)$ will converge towards the optimal objective function value, which is p^* ok. And the dual variable will converge towards the optimal dual variable. So, these are the convergence results of ADMM ok.

(Refer Slide Time: 20:30)

Stopping criteria

- Stop when primal and dual residuals small:

$$\|r^k\|_2 \leq s^{\text{primal}} \text{ and } \|s^k\|_2 \leq s^{\text{dual}}$$

Hence, $\|r^k\|_2 \rightarrow 0$ and $\|s^k\|_2 \rightarrow 0$ as $k \rightarrow \infty$

Tolerance

IIT KHARAGPUR NPTEL ONLINE CERTIFICATE

So, then the question is if this is the convergence, so how should when should you stop iterating. So, the iterations that are given to you, when should you stop those iterations. And the prescription is that you define a small number epsilon prime sorry epsilon primal and epsilon dual. So, these are basically some tolerance values, because so just like for any convergence of any optimization values, they may not be exactly equal to 0 ok. So, you show some tolerance values.

And whenever the norm of your primal residual is less than the primal tolerance, and the norm of your dual residual is less than the dual tolerance value you can stop iterating further ok. So, this is the recipe for the convergence of or for the stopping of the ADMM iterations ok.

(Refer Slide Time: 21:42)

Observations

- x - update requires solving an optimization problem

$$\min_x f(x) + \frac{\rho}{2} \|Ax - v\|_2^2$$

with, $v = Bz^k - c + u^k$

- Similarly for z-update.
- Sometimes has a closed form.
- ADMM is a meta optimization algorithm.

The slide includes a footer with logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION, and a small video inset of a presenter.

Now, we come to the main idea that you now look at the so if we go back to our ADMM ok, you see that this is an this entire thing is an optimization algorithm. But, these two steps are themselves optimization problem ok. So, so this is a typical case with the ADMM algorithm that it is called a meta optimization algorithm ok.

So, yeah so the x updates requires solving an optimization problem. Similarly, the z update require solving another optimization problem ok, but as we shall see for most of the problems that we will be looking at the z update may have a closed form solutions. Even though it is an optimization problem, it may have a close form solution. So, hence ADMM is a meta optimization algorithm. So, it is an optimization algorithm, which uses other optimization algorithms as its components.

(Refer Slide Time: 23:30)

Decomposition

- If f is separable:

$$f(x) = f_1(x_1) + \dots + f_N(x_N), \quad x = (x_1, \dots, x_N)$$
- A is conformably block separable; i.e. $A^T A$ is block diagonal.
- Then, x -update splits into N parallel updates of x_i .

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION

If your f is separable that is, if your f of x the original function if yeah. So, if f is separable, then A is conformably block separable that is if that is if you are A transpose A is block diagonal, then the x update splits into N parallel updates for x_i so yeah.

(Refer Slide Time: 24:06)

Consensus Optimization

- Problem:

$$\min_x f(x) = \sum_{i=1}^N f_i(x)$$
- ADMM form:

$$\min_{x_i, z} \sum_{i=1}^N f_i(x_i)$$

$$\text{s.t. } x_i - z = 0, \quad i = 1, \dots, N$$
- Augmented Lagrangian:

$$L_p(x_1, \dots, x_N, z, y) = \sum_{i=1}^N (f_i(x_i) + y_i^T(x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2)$$

Handwritten notes: $L(x) = \sum_{i=1}^M l_i(x_i)$, $\text{s.t. } x_i = z$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION

So, so we will we will explain that with an example. So, we take the example of now the distributed optimization problem. So, if you remember, our objective was exactly of this form ok. So, our objective was that you had to minimize summation over i is equal to 1 to M loss function of x_i , this was the total loss of x ok.

Now, this was equivalent to saying you want to minimize summation over i is equal to 1 to M $\|x_i\|$ subject to the constraint that all of these x_i 's are equal to z ok. So, this is what we have written here ok. And if you see, the augmented Lagrangian for this problem is something like this ok. So, here the x variable is basically split into N many x variables.

(Refer Slide Time: 25:37)

Consensus Optimization

- ADMM algorithm:

$$x_i^{k+1} = \operatorname{argmin}_{x_i} \left(f_i(x_i) + y_i^{kT} (x_i - z^k) + \frac{\rho}{2} \|x_i - z^k\|_2^2 \right) \rightarrow N$$

$$z^{k+1} = \frac{1}{N} \sum_{i=1}^N \left(x_i^{k+1} + \frac{1}{\rho} y_i^k \right) \leftarrow \text{Consensus}$$

$$y_i^{k+1} = y_i^k + \rho (x_i^{k+1} - z^{k+1})$$
- Final solution is z^k .

Handwritten notes: $i=1 \dots N$ and Consensus

And if that is the case, as we shall see yeah so basically now the three steps of ADMM become something like this that. You have your x update being split into so N x updates. So, this is for i is equal to 1 to N ok. So, your x update is split into N x updates ok.

Your z update z is the consensus variable ok, so z is the consensus variable ok. Your z update becomes something like this which is a close form solutions. So, remember in the general ADMM this z was also supposed to be so this z^{k+1} was supposed to be an optimization problem. But, it turns out that if you try to solve that optimization problems here. If you see this if you try to so we can write here.

(Refer Slide Time: 26:49)

Consensus Optimization

- ADMM algorithm:
$$x_i^{k+1} = \operatorname{argmin}_{x_i} (f_i(x_i) + y_i^{kT} (x_i - z^k) + \frac{\rho}{2} \|x_i - z^k\|_2^2)$$
$$z^{k+1} = \frac{1}{N} \sum_{i=1}^N (x_i^{k+1} + \frac{1}{\rho} y_i^k)$$
$$y_i^{k+1} = y_i^k + \rho (x_i^{k+1} - z^{k+1})$$
- Final solution is z^k .

Handwritten note: $z^{k+1} = \min_z \mathcal{L}_\rho(z, y^k)$

IIT KHARAGPUR | NPTEL ONLINE CERTIF

So, your z^{k+1} was nothing but minimum over z , and then your L rho, and then your x^{k+1} z y^k ok. And you see the way things are ok. So, if you minimize it like this with respect to z , you will get that you can see that it will just be the squared error. And this will be minimized, when you get you just set z to be equal to this value.

So, you can check this just by differentiating and equating it to 0 ok. And y update is of course like this ok. So, this is your ADMM iteration. And actually you can further simplify this, because now what you can do is you can just substitute this z back to here ok.

(Refer Slide Time: 28:12)

Consensus Optimization

z-update can be written as:

$$z^{k+1} = \bar{x}^{k+1} + \frac{1}{\rho} \bar{y}^{k+1}$$

Averaging the y-updates:

$$\bar{y}^{k+1} = \bar{y}^k + \rho(\bar{x}^{k+1} - z^{k+1})$$

Substituting first into second: $\bar{y}^{k+1} = 0$. Hence $z^k = \bar{x}^k$.

Revised algorithm:

$$\begin{cases} x_i^{k+1} = \operatorname{argmin}_{x_i} (f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + \frac{\rho}{2} \|x_i - \bar{x}^k\|_2^2) \\ y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1}) \end{cases} \rightarrow \bar{z}^k = \bar{x}^k = \frac{1}{N} \sum_{i=1}^N x_i^k$$

Final solution is z^k .

So, if you do that if you do the substitution, what you see is that your updates, so you can just so you can so your y update just become something like this. So, what is your y update now? Your y update is whatever are the updated x values ok. If you take the average of those if you take the average of updated x values ok, and then if you take x i minus that and then rho and at that to y k that is your y updates.

So, you have basically eliminated the z by substituting this z k plus 1 into all the updates; so, z k and z k plus 1 into all the updates ok. So, you see that into the first you get this is equal to 0, and hence you get this update. And then finally, when you substitute z k which is which becomes equal to basically x k bar, and that you substitute here you get the modified x updates.

So, your z is basically nothing but x bar of k ok. So, your z k becomes equal to x bar of k ok, where x bar of k is nothing but your 1 by N summation over i is equal to 1 to N x i k ok. So, for this kind of a consensus optimization problem so for the consensus optimization problem, which is this problem ok, which is this problem the updates are something like this ok.

(Refer Slide Time: 30:30)

Consensus Optimization

z-update can be written as:

$$z^{k+1} = \bar{x}^{k+1} + \frac{1}{\rho} \bar{y}^{k+1}$$

Averaging the y-updates:

$$\bar{y}^{k+1} = \bar{y}^k + \rho(\bar{x}^{k+1} - z^{k+1})$$

Substituting first into second: $\bar{y}^{k+1} = 0$. Hence $z^k = \bar{x}^k$.

Revised algorithm:

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}} \left(f_i(x_i) + y_i^{kT} (x_i - \bar{x}^k) + \frac{\rho}{2} \|x_i - \bar{x}^k\|_2^2 \right)$$

$$y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1})$$

Final solution is z^k .

So, these are the final updates ok. And the final solution is going to be z^k , which is nothing but \bar{x}^k . So, the average of all the x_i 's the final solution that you will get for the original problem. And as you can see this is very easily implemented in a distributed setting. So, all you have to do is you have to store the y_i^k in the central machines. And x_1^k like this till x_N^k in the machines.

And then the in the, for the first step, you have to send all the x_i 's to the local this thing is to the local machines. So, in the local machine it can compute \bar{x}^k , and then the \bar{x}^k can be sent back to individual machine. So, from the central machine the individual machine gets \bar{x}^k ok.

And then you have all the information to compute this objective function, because you have the f_i . So, f_i is on the i th machine ok. And you have the y_i^k , so y_i^k also has to be sent ok, but you have the y_i^k and you have the \bar{x}^k ok. So, then you can just optimize with respect to x_i to get x_i^{k+1} . And once you get x_i^{k+1} here, you can send that to the central machine to compute the \bar{x}^{k+1} in the central machine. And once you compute the \bar{x}^{k+1} in the central machine, you can compute all the y_i^{k+1} in the central machine, and that you can send back, so that completes the description of the algorithm.

(Refer Slide Time: 32:54)

Loss minimization

► Problem:

$$\min_x \ell(Ax - b) + r(x)$$

► Partition A and b by rows:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix},$$

where, $A_i \in \mathbb{R}^{m \times m}$ and $b_i \in \mathbb{R}^m$

► ADMM formulation:

$$\min_{x_i, z} \sum_{i=1}^N \ell(A_i x_i - b_i) + r(z)$$

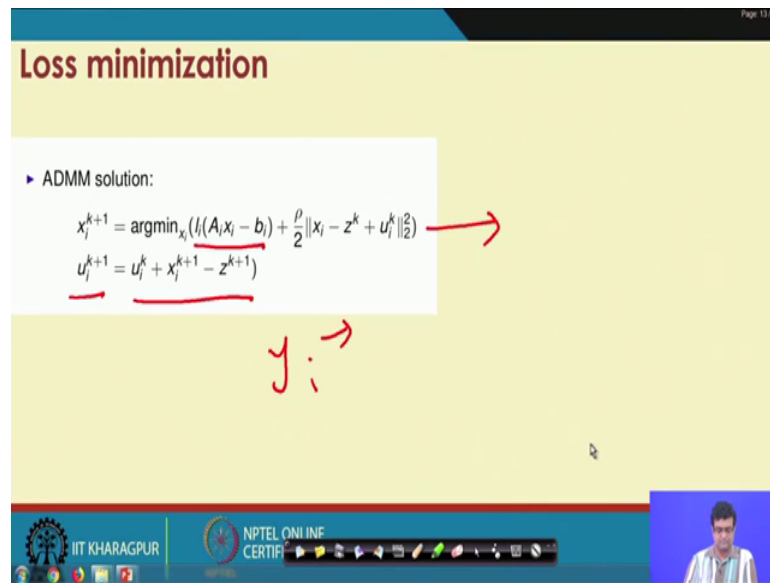
s.t.: $x_i - z = 0, i = 1, \dots, N$

loss i-th machine.
 $x_i = z$

So, we further describe, so how will you solve this or how will you use this technique to solve the distributed loss minimization problem. So, so this is our distributed loss minimization problems. So, this is a general form of our distributed loss minimization problem. You have the loss function ℓ , which is a decomposable function over this x 's ok, and rather which is a decomposable function yeah over x 's.

And you have the regularize which is dependent only on x ok. So, basically this each part ℓ_i of $A_i x_i - b_i$ is the loss on the i th machine. So, this is the loss on the i th machine, and all of the x_i should be equal to z ok. This is the constraint, so that x_i is equal to x_z . So, all of them are equal to z . So, this is the consensus constraint.

(Refer Slide Time: 34:24)



Loss minimization

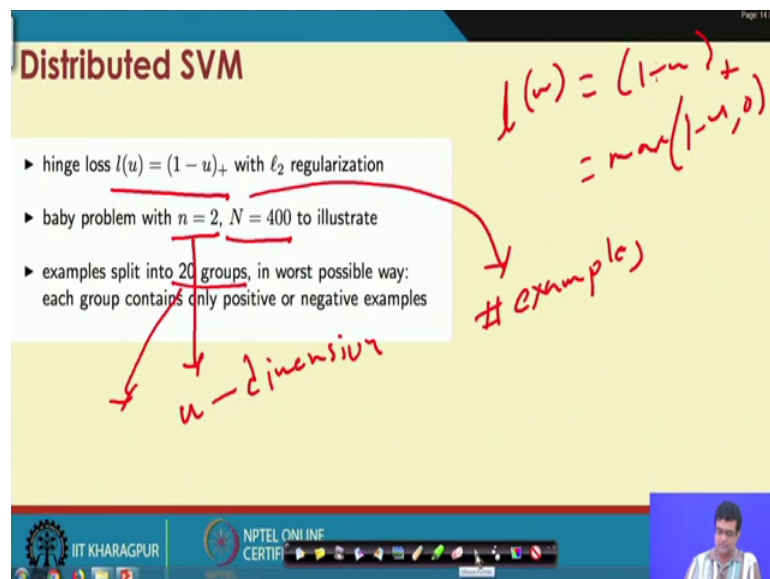
▶ ADMM solution:

$$x_j^{k+1} = \underset{x_i}{\operatorname{argmin}} (l(A_i x_i - b_i) + \frac{\rho}{2} \|x_i - z^k + u_i^k\|_2^2)$$
$$u_j^{k+1} = u_j^k + x_j^{k+1} - z^{k+1}$$

Handwritten annotations: A red arrow points from the second term of the first equation to the right. A red arrow points from the second equation down to the text "y_i".

So, as we have already discussed a little bit of arithmetic shows that this is your update equation for x_i ok. So, basically this is the last part. And your u_i is what is called the scale dual variables ok. So, your u_i is the scale dual variables, these are something like y_i ok. And then you can compute these as you can compute the using this formula. And then you can just update u_i $k+1$ ok. And then this is the final solution ok.

(Refer Slide Time: 35:23)



Distributed SVM

- ▶ hinge loss $l(u) = (1 - u)_+$ with ℓ_2 regularization
- ▶ baby problem with $n = 2$, $N = 400$ to illustrate
- ▶ examples split into 20 groups, in worst possible way: each group contains only positive or negative examples

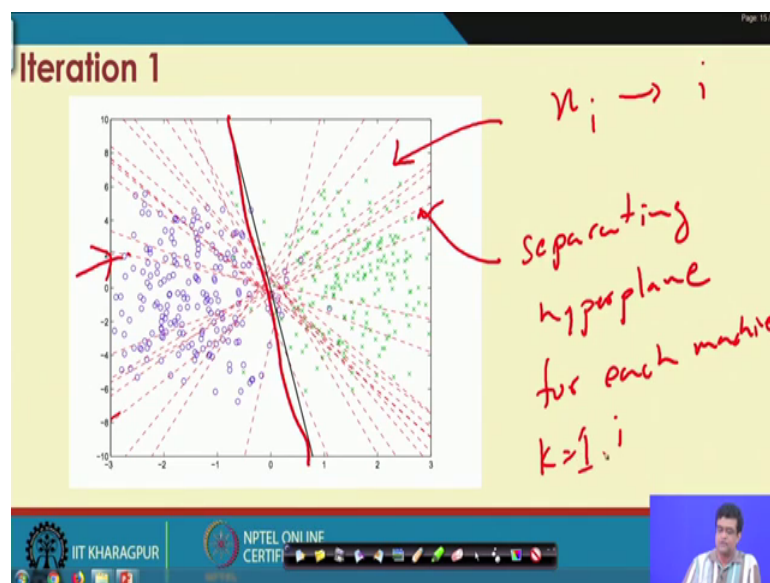
Handwritten notes: The hinge loss function is defined as $l(u) = (1 - u)_+ = \max(1 - u, 0)$. An arrow points from the text "examples split into 20 groups" to the handwritten note "u-dimensional". Another arrow points from the definition of $l(u)$ to the handwritten note "# example".

So, this is a simple small toy problem, which is taken from Stephen Boyd's slides ok. So, basically let us say that this loss function is hinge loss, which is basically your l of u is

nothing but 1 minus sorry your 1 minus u the positive of this. This is nothing but max over 1 minus u comma 0 ok.

And you have capital N is equal to 400 and small n which is the dimension. So, this is the dimension of u so dimension of u ok. So, this is basically u's dimension ok. And this is the number of examples, so this is the number of examples ok. And the examples are split into 20 groups ok, so this is the scenario. And this is the problem that we are solving ok.

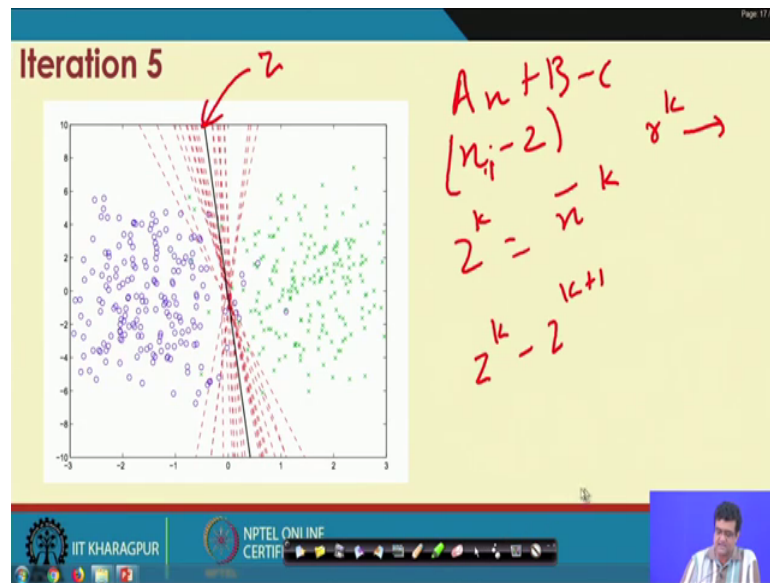
(Refer Slide Time: 37:05)



So, you see how so each of these each of these lines is the separator or so this is a classification problem. There are two classes; you can see that your blue circles are point for class 1. And green crosses are points for class 2. And this is supposed to be your optimal solution ok.

But, you have now 20 optimal solution, because you remember you have x_i for each i ok. So, and this dotted red lines are plotting the separating hyper plane. So, these are plotting the separating hyper plane for each machine i ok. And this is the iteration 1. So, your k is equal to 1 ok.

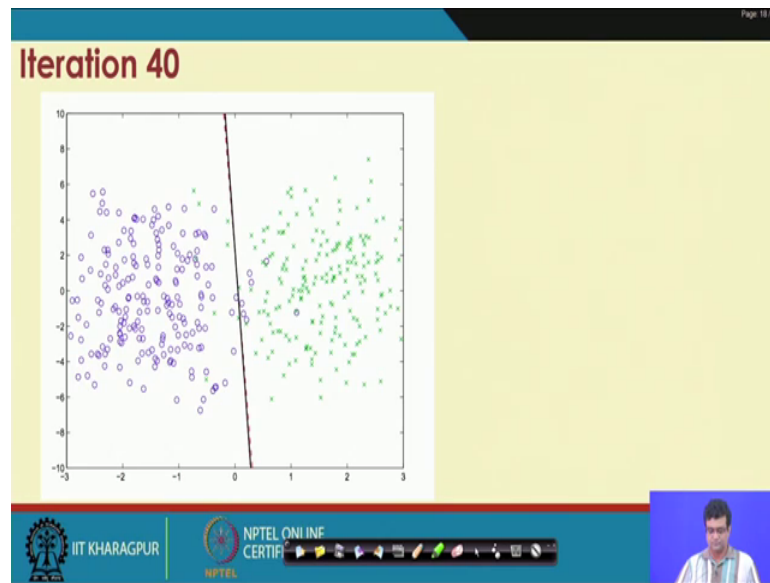
(Refer Slide Time: 38:28)



And now as the iterations progress, so for example, after 5 iterations you see that your hyper planes have all come together a little bit. So, you would remember that you are so in this case basically you are if you recall, your primal residual is something was something like $Ax + B - c$.

So, in this case, it will just be $x - z$ or $x_i - z$ ok. So, for each i your primal residual will be $x_i - z$ ok. And you can see that your this you are this black line corresponds to something like z , because your z is nothing but your \bar{x} . So, your z^k is something like \bar{x}^k ok. And the question so basically you are residuals are how close are these lines to this black line ok. And you see that they have become much more closer. So, your r^k primal residual is actually going down ok. And your dual residual is nothing but the difference between z^k and z^{k+1} .

(Refer Slide Time: 39:57)



So, you see that as the iterations are progressing your z_k and z_{k+1} are roughly the same. So, you see the black line is this z_k . So, roughly they are the same, and yeah.

(Refer Slide Time: 40:04)

So, and now you see, so all the hyper plane has converge to the black line. And the black line between two successive iterations have also become same; so, both these residuals sorry; so both the residuals have become 0 or very close to 0 at iteration 40 ok. So, this is how the algorithm progresses ok. So, with that we conclude our discussion of ADMM. And these are the references. So, the first one is the primary reference from where all the

materials have been taken. And the second one is an application that we have explored for ADMM yeah. So, the first one is the primary reference ok.

Thank you.